What is legacy?

What is legacy?

- Not written by us

What is legacy?

~~Not written by us~~

What is legacy?

~~Not written by us~~

No architecture or bad architecture

What is legacy?

~~Not written by us~~
No architecture or bad architecture
No structure or a code mess

What is legacy?

~~Not written by us~~
No architecture or bad architecture
No structure or a code mess
WTF per hour > 9000

How to measure the legacy?

How to measure the legacy?

$$L = T * H$$ [1]

[1] T - hours of tech debt, H - developer’s rate per hour

How to measure T?

What is technical debt?

What is technical debt?

- Errors and vulnerabilites

What is technical debt?

- Errors and vulnerabilites
- Maintenability problems

What is technical debt?

- Errors and vulnerabilites
- Maintenability problems
- Code duplications

What is technical debt?

- Errors and vulnerabilites
- Maintenability problems
- Code duplications
- Low test coverage

Measurement tools

SonarQube

Bugs & Vulnerabilities

960 E
Bugs

85 E
Vulnerabilities

Code Smells

122d A
Debt

8.9k
Code Smells

started 3 минуты назад

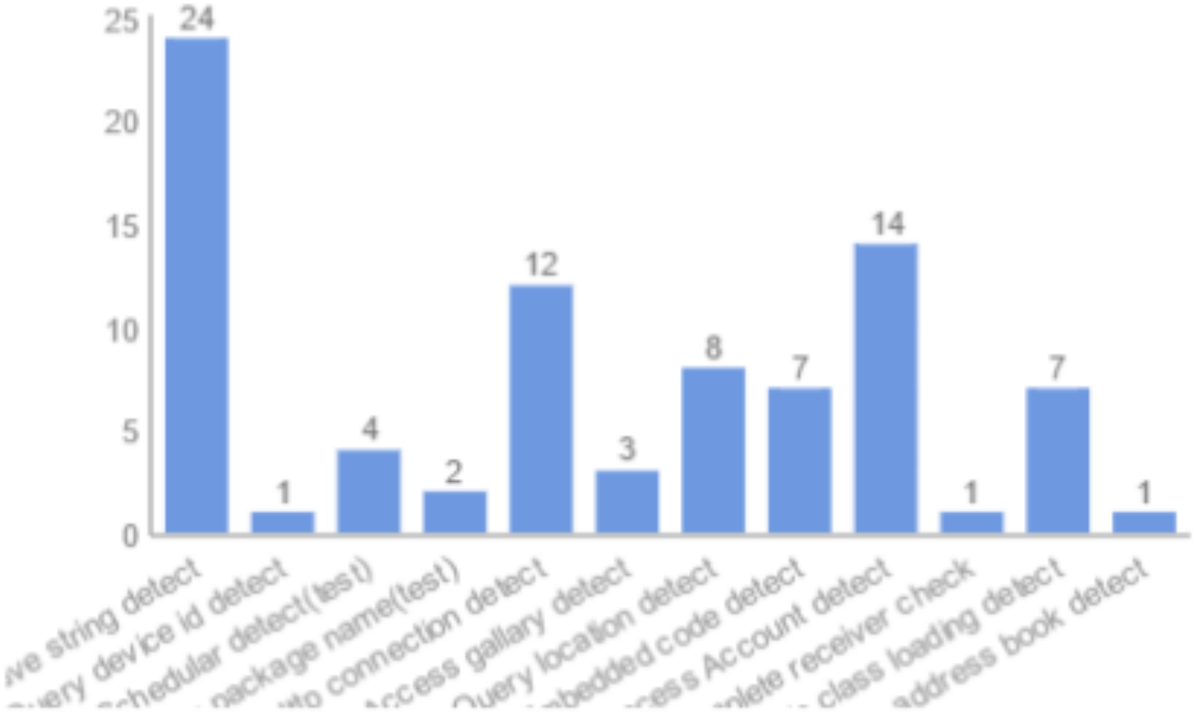
Duplications

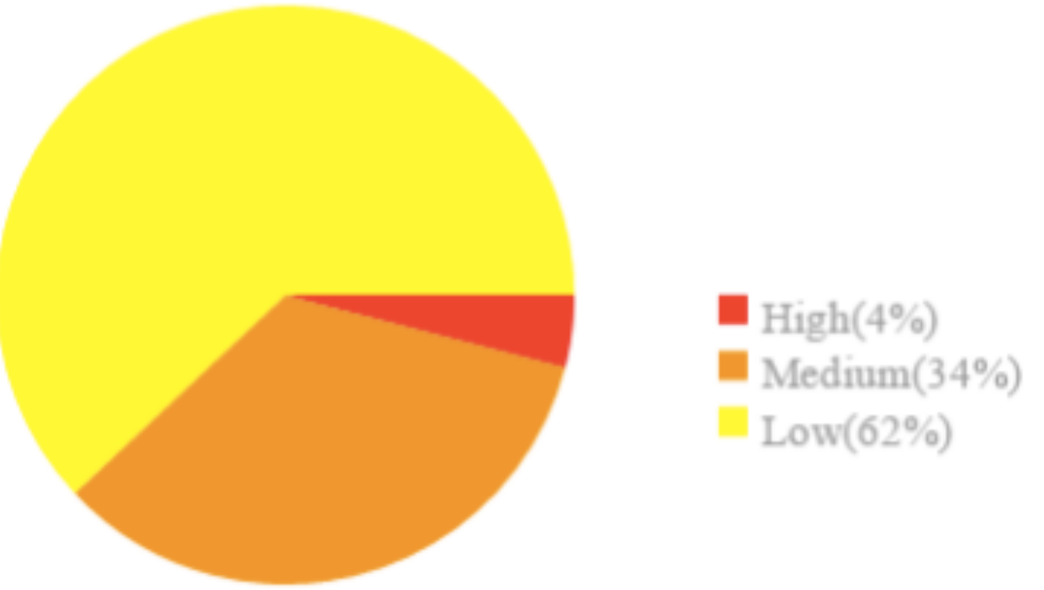
4.8%
Duplications

620
Duplicated Blocks

Measurement tools

SonarQube
MobSF/Akana/Whatever Security Analysis tool


25
20
15
10
5
0
24
1
4
2
12
3
8
7
14
1
7
1
ve string detect
uery device id detect
chedular detect(test)
package name(test)
to connection detect
Access gallary detect
Query location detect
bedded code detect
ess Account detect
plete receiver check
class loading detect
address book detect


High(4%)
Medium(34%)
Low(62%)

Measurement tools

SonarQube
MobSF/Akana/Whatever Security Analysis tool
Architecture analysis tool

Measurement tools

SonarQube
MobSF/Akana/Whatever Security Analysis tool
~~Architecture analysis tool~~

What’s an architecture?

The set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both. [2]

[2] Software Architecture in Practice, 3rd Edition, Bass, Clements, Kazman, Addison-Wesley

What the hell it means?

What the hell it means?

- Architecture is a key to the system properties, an end user is concerned about

What the hell it means?

- Architecture is a key to the system properties, an end user is concerned about
- Or product owner thinks so

What the hell it means?

- Architecture is a key to the system properties, an end user is concerned about
- Or product owner thinks so
- There are no bad or good architectures; there are ones that fit the target system properties

When you need to think about an architecture?

When you need to think about an architecture?

- No need to if you try the technology

When you need to think about an architecture?

- No need to if you try the technology
- No need to if you create a one-time script

When you need to think about an architecture?

- No need to if you try the technology
- No need to if you create a one-time script
- You definitely need to in all the other cases

What are our target properties?

What are our target properties?

- Performance

What are our target properties?

- Performance
- Localization

What are our target properties?

- Performance
- Localization
- Offline work

What are our target properties?

- Performance
- Localization
- Offline work
- Testability

What are our target properties?

- Performance
- Localization
- Offline work
- Testability
- Extendability

Testability

Testability

- Arguments and return values are explicit

Testability

- Arguments and return values are explicit
- The dependencies are explicit

Testability

- Arguments and return values are explicit
- The dependencies are explicit
- The dependencies are changeable in tests

Extendability

Extendability

- The changes are inevitable

Extendability

- The changes are inevitable
- How much do the changes cost:
  - How much do the changes-making mechanism cost?
  - How much do the change making without the mechanism cost?

Extendability

For N similiar changes approx. equation is:

$$N * ChangeCost <= MechanismCost + N * (Change2Cost)$$

What are our target properties?

What are our target properties?

- Performance

What are our target properties?

- Performance
- Localization

What are our target properties?

- Performance
- Localization
- Offline work

What are our target properties?

- Performance
- Localization
- Offline work
- Testability

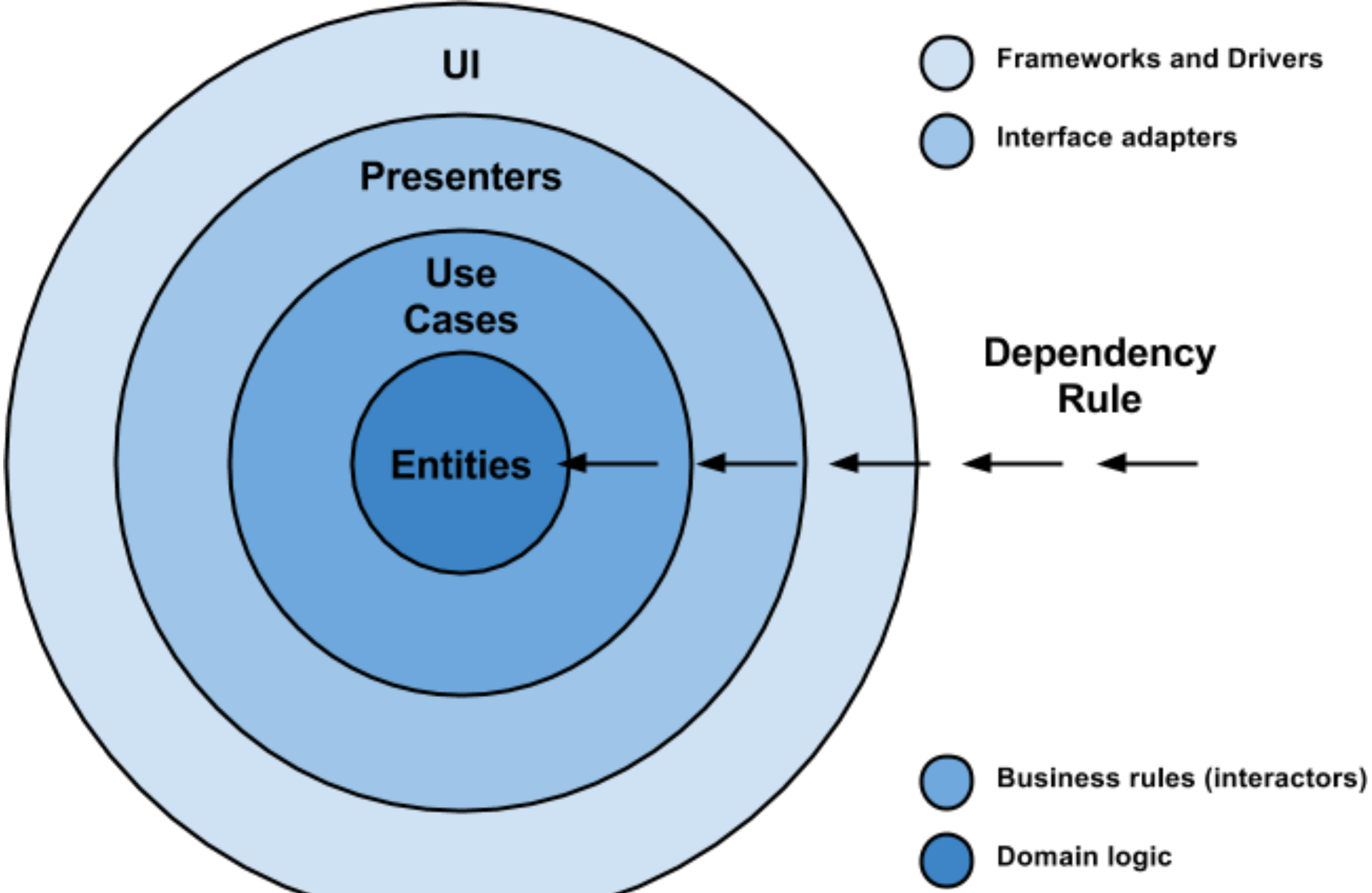
What are our target properties?

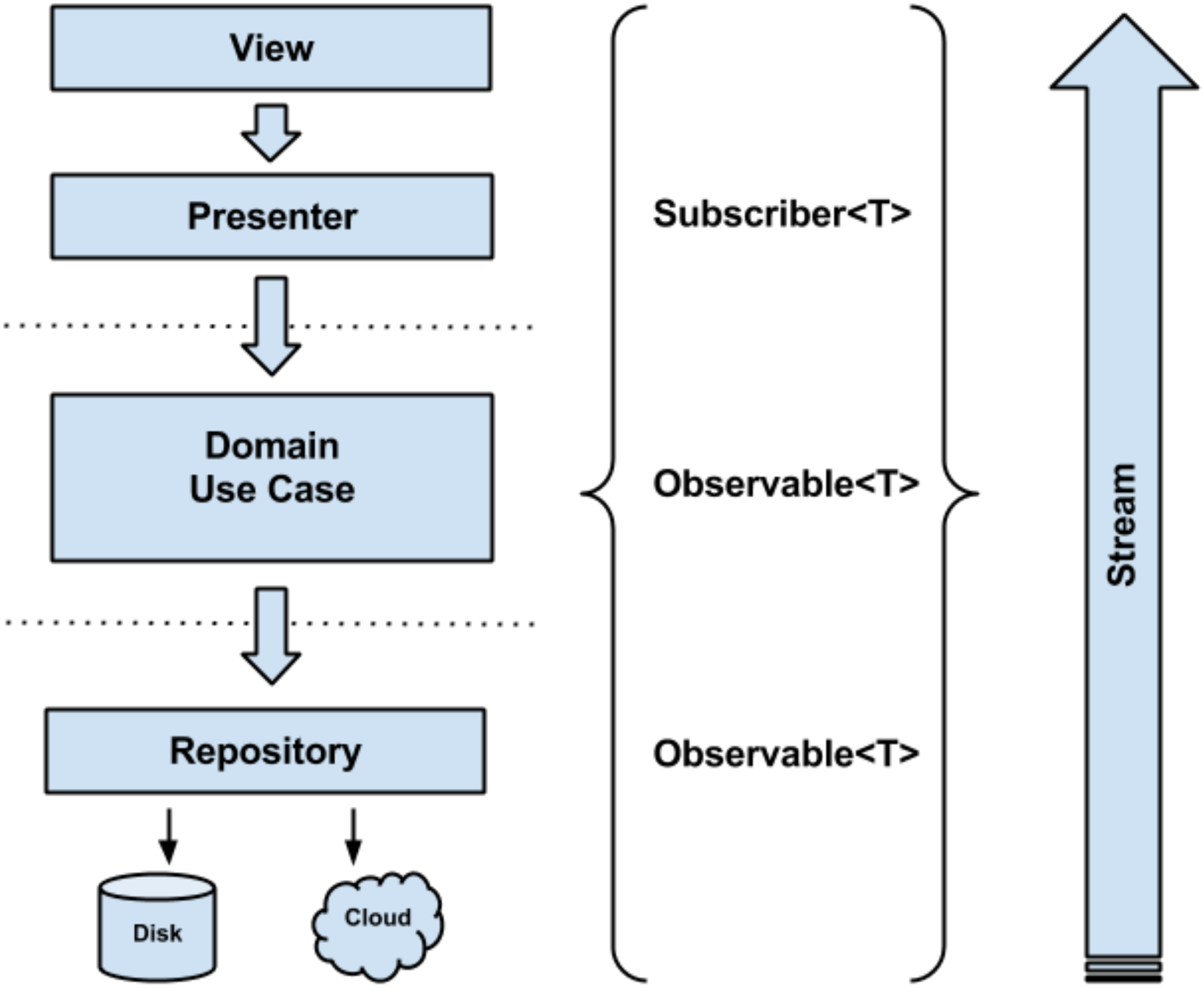
- Performance
- Localization
- Offline work
- Testability
- Extendability

What architecture provides those properties?

What architecture provides those properties?

- Clean architecture


UI
Presenters
Use Cases
Entities
Frameworks and Drivers
Interface adapters
Dependency Rule
Business rules (interactors)
Domain logic


View
Presenter
Domain
Use Case
Repository
Disk
Cloud
Subscriber<T>
Observable<T>
Observable<T>
Stream

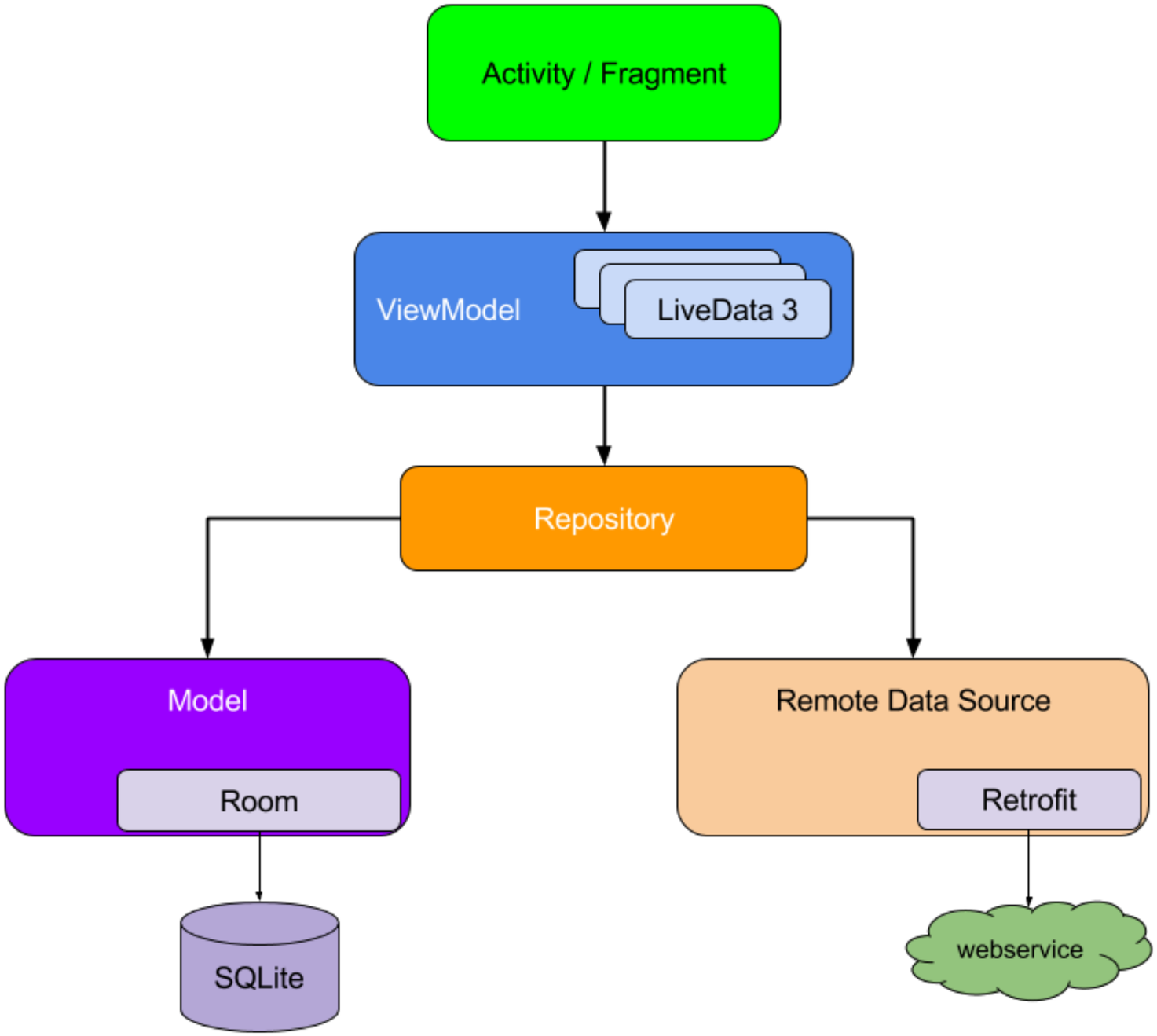
What architecture provides those properties?

What architecture provides those properties?

- Clean architecture

What architecture provides those properties?

- Clean architecture
- Google Android architecture


Activity / Fragment
ViewModel
LiveData 3
Repository
Model
Room
SQLite
Remote Data Source
Retrofit
webservice

What architecture provides those properties?

What architecture provides those properties?

- Clean architecture

What architecture provides those properties?

- Clean architecture
- Google Android architecture

What architecture provides those properties?

- Clean architecture
- Google Android architecture
- VIPER

What’s in common?

What’s in common?

In strictly layered architectures the current layer may access only the one layer below [3]

[3] Software Architecture in Practice, 3rd Edition, Bass, Clements, Kazman, Addison-Wesley

Пример с GreenDao

Пример с GreenDao

1. 4 databases

Пример с GreenDao

1. 4 databases
2. 30 tables

Пример с GreenDao

1. 4 databases
2. 30 tables
3. 278 classes for data access

Пример с GreenDao

1. 4 databases
2. 30 tables
3. 278 classes for data access
4. 20,000 LOC

Пример с GreenDao

1. 4 databases
2. 30 tables
3. 278 classes for data access
4. 20,000 LOC
5. 0% test coverage in fact

Entity example

```
public class Entity {

    private String name;
    private Long id;

    …

    public String getName();
    …

}
```

Entity request example

```
@AutoFactory
public class GetEntityRequest extends SqlRequest {

    @Override
    protected String[] getColumns() {
        return new String[] {
            F + “.” + ID_FIELD + “ AS “ + F_ID_FIELD,
            F + “.” + NAME_FIELD + “ AS “ + F_NAME_FIELD,
            …
            STORAGE + “.” + ID_FIELD + “ AS “ + STORAGE_ID_FIELD,
            …
            SYNC + “.” + ID_FIELD + “ AS “ + SYNC_ID_FIELD,
            …

        }
    }
}
```

Entity request example

```
@Override
public Entity createEntity() {
    if (!cursor.moveToFirst()) {
        return null;
    }

    …
    String name = cursor.getString(cursor.getColumnIndex(F_NAME_FIELD));
    …
    return new Entity(id, name, …);
}
```

Entity request example

```
@AutoFactory
public class GetEntityRequest extends SqlRequest {
    …
}
```

Entity request example

```
@AutoFactory
public class GetEntityRequest extends SqlRequest {
    …
}

@Generated
public class GetEntityRequestFactory {
    public GetEntityRequest create() {
        return new GetEntityRequest();
    }
}
```

Request call

```
public class MyEntityActivity extends Activity {

    @Inject GetEntityRequestFactory requestFactory;

    @Override
    protected void onCreate(Bundle instance) {
        super.onCreate(instance);
        Entity entity = requestFactory.create().createEntity();
        show(entity);
    }
}
```

What are the issues?

What are the issues?

1. Disk access on the UI thread

What are the issues?

1. Disk access on the UI thread
2. A lot of boilerplate

What are the issues?

1. Disk access on the UI thread
2. A lot of boilerplate
3. Presenation layer accesses the data hardening the unit-testing

What are the issues?

1. Disk access on the UI thread
2. A lot of boilerplate
3. Presenation layer accesses the data hardening the unit-testing
4. The data model is unknown

Extending the layer

Extending the layer

1. Separate module

Extending the layer

1. Separate module
2. Unit-test covered 100%

Why?

Why?

1. Getting to know the data model

Why?

1. Getting to know the data model
2. Providing guarantees it works

Why?

1. Getting to know the data model
2. Providing guarantees it works
3. Protect of incorrect changes

Getting to know the data model

Getting to know the data model

Stetho [4]

[4] http://facebook.github.io/stetho/

Getting to know the data model

Stetho [4]
Db files

[4] http://facebook.github.io/stetho/

Getting to know the data model

Stetho [4]
or Android file Explorer
Db files
sqldelight [5]

---

[4] http://facebook.github.io/stetho/

[5] https://github.com/square/sqldelight

Stetho

```
compile 'com.facebook.stetho:stetho:1.5.0'
```

Stetho

```
public class MyApplication extends Application {
  …
  public void onCreate() {
    super.onCreate();
    …
    Stetho.initializeWithDefaults(this);
  }
}
```

Elements Network Sources Timeline Profiles Resources Audits Console

- Frames
- Web SQL
  - apod.db
    - android_metadata
    - rss_items
    - sqlite_sequence
- IndexedDB
- Local Storage
- Session Storage
- Cookies
- Application Cache

```
> SELECT * FROM rss_items;
```

_…	title	description_image_u…	description_text
43	Our Galaxys Magneti…	http://antwrp.gsfc.…	What does the magne…
44	The Milky Way over …	http://antwrp.gsfc.…	You may have heard …
45	A Twisted Solar Eru…	http://antwrp.gsfc.…	A Twisted Solar Eru…
46	Light from Cygnus A	http://antwrp.gsfc.…	Celebrating astrono…
47	Interior View	http://antwrp.gsfc.…	Interior View
48	Launch to Lovejoy	http://antwrp.gsfc.…	Launch to Lovejoy
49	The Complex Ion Tai…	http://antwrp.gsfc.…	What causes the str…

```
> SELECT _id, title FROM rss_items WHERE description_text CONTAINS
'%comet%';
near "CONTAINS": syntax error (code 1): , while compiling: SELECT
_id, title FROM rss_items WHERE description_text CONTAINS '%comet%';
> SELECT _id, title FROM rss_items WHERE description_text LIKE
'%comet%';
```

_id	title
49	The Complex Ion Tail of Comet Lovejoy

```
> PRAGMA user_version;
```

user_version
1

```
>
```

Device File Explorer

Samsung SM-G930F Android 7.0, API 24

Name	Permissions
acct	dr-xr-xr-x
cache	drwxrwx---
config	drwxr-xr-x
cpefs	drwxrwx--x
d	lrwxrwxrwx
data	drwxrwx--x
app	drwxrwx--x
data	drwxrwx--x
android	drwxrwx--x
com.android.apps.tag	drwxrwx--x
com.android.backupconfirm	drwxrwx--x
com.android.bluetooth	drwxrwx--x
com.android.bluetoothmidiservice	drwxrwx--x
com.android.bookmarkprovider	drwxrwx--x
com.android.calendar	drwxrwx--x
run-as: Could not set capabilities: Operation not permitted	
com.android.calllogbackup	drwxrwx--x

sqldelight

```
buildscript {
  repositories {
    mavenCentral()
  }
  dependencies {
    classpath 'com.squareup.sqldelight:gradle-plugin:0.6.1'
  }
}

apply plugin: 'com.squareup.sqldelight'
```

sqldelight

```
CREATE TABLE entity (
  _id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL
);
```

sqldelight

```
public class Entity {

  private Long id;

  private String name;

  public Long getId() { … }
  public String getName() { … }
);
```

sqldelight

```
@Table(name = “Entity”)
public class Entity {

  @Column(name = “_id”)
  private Long _id;

  @Column(name = “NAME”)
  private String name;

  public Long getId() { … }
  public String getName() { … }
);
```

Integration into the app - CIDR!

Integration into the app - CIDR!

1. Create unit test

Integration into the app - CIDR!

1. Create unit test
2. Implement

Integration into the app - CIDR!

1. Create unit test
2. Implement
3. Deprecate

Integration into the app - CIDR!

1. Create unit test
2. Implement
3. Deprecate
4. Replace

Integration into the app - CIDR!

Integration into the app - CIDR!

1. Create unit test

Create unit test

```
@Override
protected void onCreate(Bundle instance) {
    super.onCreate(instance);
    Entity entity = requestFactory.create().createEntity();
    show(entity);
}
```

Create unit test

```
@Override
protected void onCreate(Bundle instance) {
    super.onCreate(instance);
    Entity entity = requestFactory.create().createEntity();
    show(entity);
}
```

Create unit test

```
@Override
protected void onCreate(Bundle instance) {
    super.onCreate(instance);
    GetEntityRequest request = requestFactory.create();
    Entity entity = request.createEntity();
    show(entity);
}
```

Create unit test

```
@VisibleForTesting
protected Entity loadEntity() {
    return requestFactory.create().createEntity();
}
```

Create unit test

```
@VisibleForTesting
protected Observable<Entity> loadEntity() {
    return Single.just(
        requestFactory.create().createEntity()
    );
}
```

Create unit test

```
@Override
protected void onCreate(Bundle instance) {
    super.onCreate(instance);
    loadEntity()
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread());
        .subscribe({entity, throwable} -> {
            show(entity);
        });

}
```

Create unit test

```
@Override
protected void onCreate(Bundle instance) {
    super.onCreate(instance);
    viewModel.loadEntity()
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread());
        .subscribe({entity, throwable} -> {
            show(entity);
        });

}
```

Create unit test

```
@Test
public void shouldLoadEntity() {
    ViewModel viewModel = new ViewModel();

    viewModel.loadEntity()
        .subscribe({entity, throwable} -> {
            checkEntity(entity);
        });

}
```

Create unit test

```
public void checkEntity(Entity entity) {
    assertEquals(NAME, entity.getName());
    assertEquals(ID, entity.getId());
    …
}
```

Integration into the app - CIDR!

Integration into the app - CIDR!

1. Create unit test

Integration into the app - CIDR!

1. Create unit test
2. Implement

Implement

```
public interface EntityDAO {

    Observable<Entity> loadEntity();
}
```

Implement

```
public class ViewModel {

    @Inject EntityDao entityDao;

    public Observable<Entity> loadEntity() {
        return entityDao.loadEntity();
    }
}
```

Implement

```
public class GreenDaoEntityDAO implements EntityDAO {

    public Observable<Entity> loadEntity() {
        RxDao<DbEntity> rxDao = daoSession.getEntityDao().rx();
        return rxDao.load();
    }
}
```

Problem №1

Problem №1

1. Database object is not convenient to use in presentation

Problem №1

1. Database object is not convenient to use in presentation
2. More likely there are Entity and DbEntity.

Problem №1

1. Database object is not convenient to use in presentation
2. More likely there are Entity and DbEntity.
3. What to do?

Converters!

Converters

```
public class GreenDaoEntityDAO implements EntityDAO {

    public Observable<Entity> loadEntity() {
        RxDao<DbEntity> rxDao = daoSession.getEntityDao().rx();
        return rxDao.load().flatMap(dbEntity -> convert(dbEntity));
    }

}
```

Converters

```
public class GreenDaoEntityDAO implements EntityDAO {

    public Observable<Entity> loadEntity() {
        RxDao<DbEntity> rxDao = daoSession.getEntityDao().rx();
        return rxDao.load().flatMap(dbEntity -> convert(dbEntity));
    }

}
```

Problem №2

Problem №2

1. Junior developers doesn’t know anything about your architecture

Imagine one needs application module data…

```
public class GreenDaoEntityDAO implements EntityDAO {

    public Observable<Entity> loadEntity() {
        String userId = ApplicationSettings.getUserId();
        RxDao<DbEntity> rxDao = daoSession.getEntityDao().rx();
        return rxDao.load(userId).flatMap(dbEntity -> convert(dbEntity));
    }

}
```

Imagine one needs application module data…

Imagine one needs application module data…

- Mode the layer into separate module

Imagine one needs application module data…

- Mode the layer into separate module
- Cover it with unit-tests

Imagine one needs application module data…

- Mode the layer into separate module
- Cover it with unit-tests
- Ship binary

Integration into the app - CIDR!

Integration into the app - CIDR!

1. Create unit test

Integration into the app - CIDR!

1. Create unit test
2. Implement

Integration into the app - CIDR!

1. Create unit test
2. Implement
3. Deprecate

Deprecate

```
/**
 * @deprecated Please, use {@link EntityDao#loadEntity()}
 */
@Deprecated
@AutoFactory
public class GetEntityRequest {
...
}
```

Integration into the app - CIDR!

Integration into the app - CIDR!

1. Create unit test

Integration into the app - CIDR!

1. Create unit test
2. Implement

Integration into the app - CIDR!

1. Create unit test
2. Implement
3. Deprecate

Integration into the app - CIDR!

1. Create unit test
2. Implement
3. Deprecate
4. Replace

Final statistics

Final statistics

1. 4 databases

Final statistics

1. 4 databases
2. 30 tables

Final statistics

1. 4 databases
2. 30 tables
3. 84 data access classes (against 278)

Final statistics

1. 4 databases
2. 30 tables
3. 84 data access classes (against 278)
4. 7400 LOC including tests (againt 20,000 without ones)

Final statistics

1. 4 databases
2. 30 tables
3. 84 data access classes (against 278)
4. 7400 LOC including tests (againt 20,000 without ones)
5. 100% test coverage in fact (было 0%)

Quality Gate Passed

Bugs & Vulnerabilities

0 A
Bugs

0 A
Vulnerabilities

Code Smells

20min A
Debt

1
Code Smells

started 5 минут назад

Duplications

0.0%
Duplications

0
Duplicated Blocks

What are the target goals reached?

What are the target goals reached?

- Performance ✅

What are the target goals reached?

- Performance ✅
- Localizable

What are the target goals reached?

- Performance ✅
- Localizable
- Offline work ✅

What are the target goals reached?

- Performance ✅
- Localizable
- Offline work ✅
- Testability ✅

What are the target goals reached?

- Performance ✅
- Localizable
- Offline work ✅
- Testability ✅
- Extendability ✅

Where to apply CIDR approach?

Where to apply CIDR approach?

- Refactoring the network layer

Where to apply CIDR approach?

- Refactoring the network layer
- Moving to MVVM

Where to apply CIDR approach?

- Refactoring the network layer
- Moving to MVVM
- Basically introducing the new layer

Where not to apply?

Where not to apply?

- Security

Where not to apply?

- Security
- Tools and utility classes

Where not to apply?

- Security
- Tools and utility classes
- Any cross-component code

About me

About me

- Vladimir Ivanov - Lead Software Engineer in EPAM

About me

- Vladimir Ivanov - Lead Software Engineer in EPAM
- More than 7 years of Android apps development

About me

- Vladimir Ivanov - Lead Software Engineer in EPAM
- More than 7 years of Android apps development
- More than 15 published applications

About me

- Vladimir Ivanov - Lead Software Engineer in EPAM
- More than 7 years of Android apps development
- More than 15 published applications
- Wide interest in mobile technologies


QA